Distinguishing tests for nondeterministic finite state machines

S. Yu. Boroday
Institute of Applied Mathematics and Mechanics,
National Academy of Sciences of Ukraine,
74, R.Luxemburg street, Donetsk, 340114, Ukraine
Tel: (038) 622 510 144
E-mail: math@iamm.ac.donetsk.ua

Abstract

In this paper, testing of deterministic implementations of nondeterministic specification FSMs is considered. Given two nondeterministic FSMs, a black box deterministic FSM is known to be a correct implementation of at least one them. We want to derive a test that determines whether this black box is a correct implementation of the first NDFSM. No upper bound on the number of states of the black box is known. The necessary and sufficient conditions for test existence are found. A method for constructing a conditional test of a minimal length is proposed. Upper bounds of multiplicity, length and overall length close to minimal are obtained.

Keywords

Finite state machine, nondeterministic finite state machine, distinguishing test, conformance testing.

1 INTRODUCTION

FSM based languages are widely used for protocol specification. Protocol conformance testing is often formalised as a problem of verifying the equivalence of a deterministic implementation to a given deterministic specification machine.

A. Petrenko et al. (eds.), *Testing of Communicating Systems*

Since most existing protocols allow alternatives and options, a nondeterministic model of specifications seems to be a more versatile model for describing protocols [Petrenko, Yevtushenko, Bochmann]. An implementation machine in this paradigm is still deterministic. The implementation FSM conforms to a given nondeterministic specification FSM (NDFSM) if the implementation machine produces an output sequence that can also be produced by the specification machine in response to every input sequence. Unlike testing of deterministic FSMs, testing of NDFSMs has not been studied sufficiently well.

Conformance testing of deterministic implementations against NDFSMs is considered in [Petrenko, Yevtushenko, Lebedev, Das]. An upper bound on the implementation's state number is assumed. In [Petrenko, Yevtushenko, Bochmann] tests distinguishing states of NDFSM were also studied. Based on these tests, a method for deriving test suites from a given NDFSM complete within the given number of states is proposed in that paper.

In [Lukjanov], the following problem is considered. A black box DFSM is known to be a correct implementation of at least one of the two NDFSMs A and B. The experimenter wants to test a black box DFSM to determine whether or not the DFSM is a correct implementation of A. No upper bound on the number of states of the black box is given. The black box can be a correct implementation of one of them only, or it can correctly implement each of them at the same time. A test that determines whether a black box is a correct implementation of the NDFSM A is called a distinguishing test. In [Lukjanov], sufficient conditions for test existence and a test derivation method for some cases of specification machines are obtained.

In this paper, we address the above problem and obtain necessary and sufficient conditions. A method for deriving a conditional test with a minimal length and multiplicity is proposed. Upper bounds on length, multiplicity and overall length are obtained. These bounds are close to minimal.

2 PRELIMINARIES

Definition 1. *A nondeterministic finite state machine (NDFSM)* is a quintuple $A=(S,X,Y,F,s_0)$ where S, X, Y are finite and nonempty sets of states, input and output symbols, respectively, s_0 is an initial state, $F: X\times Y\rightarrow 2^{(S\times Y)}$ is a behaviour function.

Let sp denote a set of states in which an NDFSM can move from state s upon input word $p\in X^*$. $s(p,q)$ denotes the set of states, in which the NDFSM can move from s upon input word $p\in X^*$ with output word $q\in Y^*$. $\lambda(s,p)$ denotes a set of all output words, which the NDFSM can produce from the state s upon input p. λ_A denotes the set of all input output words $p,\lambda_A(s_0, p)$ and is called the FSMs behaviour. A deterministic FSM R is a correct implementation of the NDFSM A if $\lambda_R\subseteq\lambda_A$. If $|s(x,y)|\leq 1$ for all $x\in X$, $y\in Y$, $s\in S$, the NDFSM is observable. If $s(x,y)=\varnothing$

for some $x \in X$, $y \in Y$, $s \in S$, the NDFSM is called partially defined, otherwise it is called completely defined. It is known that for each NDFSM an observable NDFSM with the same behaviour exists. Deterministic acceptors (Rabin Scott automata) in the alphabet $X \times Y$ can be considered as a (partially defined) NDFSM with the input alphabet X, output alphabet Y and a set of final states. So notations extend on acceptors (Rabin Scott automata) in the alphabet $X \times Y$. $L(Q)$ denotes the set of words which the automaton Q accepts.

Let us consider the notion of a testing acceptor — a special case of a conditional test. An acyclic acceptor Q in the alphabet $X \times Y$ is called a single testing acceptor if for every acceptor's nonfinal state s there exists only one x_s such that $sx_s \neq \varnothing$. The acceptor is called acyclic if $s(p,q)=s$ implies p,q are the empty words. In [Petrenko, Yevtushenko, Bochmann], a final state of the testing acceptor is called a 'fail' state. A testing acceptor corresponds to an algorithm that conducts a single adaptive experiment with the black box as follows.

1. Declare the initial state s_0 as a current state.
2. From the current state s submit an input signal x_s such that $sx_s \neq \varnothing$ to the black box.
3. Read the output signal y produced by the black box.
4. If $s(x_s, y) \neq \varnothing$, then assume state $s(x_s, y)$ as a current state and go to Step 2
5. If the current state is final, then return the result 'fail', else 'pass'.

The number of inputs x_s submitted during this test is called the length of the test.

Since the acceptor is acyclic, testing is always finite and the result is defined. It is obvious that the result is 'pass' iff $L(Q) \cap \lambda_R = \varnothing$.

Let Q_i be a maximal single testing acceptor which is a subautomaton of Q. This means that a single testing acceptor Q_i can be obtained from Q by 'erasing' some (or none) states and transitions and every Q' such that if Q_i is a subautomaton of Q' and Q' is a subautomaton Q then $Q' = Q_i$. Let $\{Q_1,...,Q_N\}$ be the set of all such acceptors. This set is obviously finite. Define an arbitrary order on acceptors.

The following test procedure corresponds to a testing acceptor Q.

1. Order acceptors in a sequence $Q_1,...,Q_N$.
2. $i := 1$, L be an empty set.
3. If there are $px,qy \in L$, $y' \in Y$ such that px,qy' is a prefix of a word from $L(Q_i)$ but px,qy is not then go to 6.
4. Reset the black box to the initial state.
5. Perform test Q_i. If the result is 'fail', go to 9.
6. $i := i + 1$.
7. if $i \leq n$ go to 1.
8. The result is 'pass'. Stop.
9. The result is 'fail'. Stop.

Step 3 is included to prevent redundant testing when the result of the test Q_i with black box is obviously 'pass'.

The maximal number of resets during testing is called the multiplicity of the test Q with the given black box. The maximal multiplicity of the test Q with an

arbitrary FSM is called the multiplicity of the test Q. The maximal length of test Q_i during testing is called the length of the test Q. The maximal length of the test with an arbitrary FSM is called the length of the test Q. The number of all inputs offered to the black box is called the overall length of the test.

The order of acceptors $Q_1,...,Q_N$ can affect the test multiplicity and length in a general case.

It is obvious that the result is 'pass' iff $L(Q)\cap\lambda_R=\varnothing$. So more sophisticated algorithms for checking the intersection of $L(Q)$ and the black box behaviour could be developed to reduce the overall length of a test.

A testing acceptor Q is called a test distinguishing the NDFSM A from the NDFSM B if the result of the test Q with every correct implementation of A is 'pass' and the result of the test with a black box which is a correct implementation of B but is not a correct implementation of A is 'fail'. Initialised, completely defined implementation and specification FSMs only are considered in this paper.

The existence of a test distinguishing A from B does not imply the existence of a test distinguishing B from A. But if both tests exist we can test whether the given black box is a correct implementation of A, B or both of them.

In the next section, we consider the problems of the existence and derivation of NDFSM distinguishing tests.

3 TEST EXISTENCE

It is known that a distinguishing test exists not for all NDFSMs. The following theorem gives necessary and sufficient conditions for the test's existence:

Theorem 1. A test distinguishing an NDFSM A from B exists if and only if $[\lambda_B \setminus \lambda_A]\cap\lambda_{\mu\cap B[}$ is a finite set.

Here $[W]$ denotes the closure of the set of words W under prefix (if $pp'\in W$ then $p\in[W]$). $]A\cap B[$ is the largest completely defined NDFSM whose behaviour is included in A and B behaviours. The FSM $]A\cap B[$ is the largest if the behaviour of any NDFSM C is included in the behaviours of A and B then NDFSM C behaviour is also included in the behaviour of $]A\cap B[$.

The conditions of Theorem 1 are constructive. Let us make some auxiliary constructions based on the state pair graph in order to prove the theorem.

1. Build an acceptor (the graph of A and B state pairs) in the alphabet $X\times Y$ with the state set $S^A\times S^B \cup \{f\}$, transitions are defined as follows:

$$(s^A,s^B)(x,y) = \begin{cases} (t^A, t^B), & \text{if } s^A(x,y)=t^A \text{ and } s^B(x,y)=t^B; \\ f, & \text{if } s^A(x,y)=\varnothing, \text{ but } s^B(x,y)\neq\varnothing. \end{cases}$$

2. $i:=0$, $V_0:=\{f\}$.
3. $i:=i+1$ and $V_i:=V_{i-1}$.
4. For $s^A\in S^A$, $s^B\in S^B$, do:

if $(s^A,s^B)\notin V_i$ and $(s^A,s^B)x \subseteq V_{i-1}$ for a certain x, then do $V_i := V_i \cup (s^A,s^B)$ and mark all transitions $((s^A,s^B),x,y)$ of the constructed automaton, where $y\in Y$.

5. If $V_i \neq \varnothing$, go to 3.
6. $V:=V_i$.
7. Delete for nonmarked transitions outgoing from all states from V, the obtained automaton with the final state f is denoted $D(A,B)$.
8. Delete all states from which f is not reachable.
9. Delete all marked transitions outgoing from V. The resulting automaton with V as the set of final states is denoted $KER(A,B)$.

In step 1, the acceptor which accepts the opening of the set of word $\lambda_B \setminus \lambda_A$ under prefix was constructed. The opening under prefix means the set of all words from $\lambda_B \setminus \lambda_A$ whose own prefix can not be found in $\lambda_B \setminus \lambda_A$. On step 2–5 all r-distinguishable state pairs are included in V. As shown in [Petrenko, Evtushenko, Bochmann] s_A and s_B are r-distinguishable if and only if there exist no correct implementation of B which is a correct implementation of A.

According to [Petrenko, Yevtushenko, Bochmann] and [Lukjanov], a testing acceptor $D(A,B)$ with the initial state $(s^A,s^B)\in V_i$ is a simple test distinguishing A and B.

The set $[\lambda_B \setminus \lambda_A] \cap \lambda_{M \cap B[}$ in Theorem 1 is the set of prefixes of $L(KER)$. Since final states of KER do not have cycles, $[\lambda_B \setminus \lambda_A] \cap \lambda_{M \cap B[}$ and $L(KER)$ are either both finite or both infinite at the same time. This implies the following theorem that is equivalent to Theorem 1.

Theorem 2. A distinguishing test for NDFSMs A and B exists if and only if $L(KER)$ is a finite set, i.e. KER is acyclic.

To prove the statement, we use the following proposition.

Proposition 1. Let $L \subseteq \lambda_A$ and $L \subseteq \lambda_C$ where A is an NDFSM, C is a DFSM and both FSMs are completely defined. Then L is included in the behaviour of a correct implementation of A.

Proof of Theorem 1. Conditions are obviously sufficient. We prove that they are necessary. Assume that $[\lambda_B \setminus \lambda_A] \cap \lambda_{M \cap B[}$ is infinite and Q is a testing acceptor distinguishing A from B. Since the language is regular, there exist such words v, u, w in the alphabet $X \times Y$ such that $vu^* \subseteq \lambda_{M \cap B[}$ but $vu^*w \subseteq \lambda_B \setminus \lambda_A$. Due to Proposition 1, there exists a correct implementation DFSM C of $]A \cap B[$, $vu^* \subseteq \lambda_C$. The DFSM C is a correct implementation of both NDFSMs A and B. Perform the test Q with C. Let the length of the test Q with the DFSM C be equal to k.

It is obvious that $\lambda_C \setminus vu^* (X \times Y)^* \cup vu^k w \subseteq \lambda_B$.

Let H be an implementation DFSM of B which includes $\lambda_C \setminus vu^* (X \times Y)^* \cup vu^k w$ in its behaviour as H. Such an implementation exists according to Proposition 1. H is not a correct implementation of A because $vu^*w \subseteq \lambda_B \setminus \lambda_A$. Thus, results of the test Q on H and C are different. Therefore, H and C have different outputs for at least one input word of length k. This contradicts the construction of H and C. The theorem has been proven.

4 TEST GENERATION

The following lemma can be proven similarly to Theorem 1.

Lemma 1. Let Q be a testing acceptor distinguishing A from B. Then every word from $L(KER)$ is a prefix of a word from $L(Q)$.

$D(A,B)$ with the initial state (s^A,s^B) from V is a minimal testing acceptor distinguishing A and B. Classes of correct implementations of states s^A, s^B do not intersect. From these facts and Lemma 1, the following theorem is obtained.

Theorem 3. If a test distinguishing NDFSM A from B exists then $D(A,B)$ is a distinguishing test of minimal length and multiplicity.

Theorem 3 obviously defines a test generation procedure. Lemmea 1 implies that the length and multiplicity of the test D in the worst case do not depend on the assumed order of simple tests $Q_1,...,Q_N$.

Several testing acceptors with various overall lengths of corresponding tests can be constructed. To find an algorithm for constructing tests of a minimal overall length we needs a more complex conditional test model than a testing acceptor. But the latter can yet be useful.

By erasing outputs in the testing acceptor graph we also can obtain an unconditional test.

It is obvious that the test length, multiplicity and overall length do not exceed nm, $|X|^{nm}$, $nm|X|^{nm}$ respectively, where n and m are state numbers of NDFSMs. These upper bounds have the same order as minimal.

5 ACKNOWLEDGEMENTS

The author thanks Alexandre Petrenko and Igor Grunsky for discussions and helpful suggestions. The author would like also to thank them and anonymous reviewers for comments that helped improve the presentation of the paper.

6 REFERENCES

Boroday, S. Yu. (1995) Experiments in classes of NDFSM implementations. Proceeding of II Ukrainian conference on automatic control 'Automatika 95', Lvov, p 101.

Lukjanov B. D. (1995) On distinguishing and checking experiments with nondeterministic automata, *Kibernetika i systemny analiz*, **5**. Plenum Publishing Corporation. New York. pp. 56-66.

Petrenko, A., Yevtushenko, N., Lebedev, A., Das, A. (1993) Nondeterministic state machines for protocol testing. *IFIP Transactions Protocol Test Systems*

VI (the Proceedings of IFIP TC6 Sixth international Workshop on Protocol Test Systems 1993) North-Holland, 1994, pp.193-208.

Petrenko, A., Yevtushenko, N., Bochmann G. (1996) Testing deterministic implementations from nondeterministic FSM specifications. *IFIP Testing of Communicating Systems (the Proceedings of IFIP TC6 9th International Workshop on Protocol Test Systems 1996)* Chapman & Hall, 1996, pp125-140.

Starke, P.H. (1972) Abstract automata. North-Holland/American Elsevier, 419p.

7 BIOGRAPHY

Boroday Sergey is an engineer of the Institute of Applied Mathematics and Mechanics (Donetsk, Ukraine), the National Academy of Sciences of Ukraine. He received his Ph.D. degree in Math from the Saratov State University (Russia) in 1997. His main research interests are in automata theory and software, hardware and protocol testing.